*Presentation Binary Clock + USART output*

Content

[Goal of the project 2](#_Toc90221320)

[Description of the solution 3](#_Toc90221321)

[2.1 Idea 3](#_Toc90221322)

[2.2 Initialization 3](#_Toc90221323)

[2.3 Interrupt Setting 4](#_Toc90221324)

[2.4 Timer Overflow Interrupt 5](#_Toc90221325)

[Results 6](#_Toc90221326)

[Conclusions 7](#_Toc90221327)

[Sources 7](#_Toc90221328)

# Goal of the project

The goal of this “low-level-programming project” is the implementation of a binary clock along with a USART Output.

It should be able to display the time according to the time that has passed since the start of the clock. Format of the clock (in binary) should be as follows: 24 hours : 60 minutes : 60 seconds.

Counting should be like follows:

Increment seconds every once a second -> When seconds = 60 seconds -> Overflow -> Increment minutes by 1 and reset seconds to 0 -> When minutes = 60 minutes -> Overflow -> Increment hours by 1 and reset minutes to 0

The requirements to this implementation are:

* Counting hours, minutes and seconds
* USART through the console
* Output to the pins (I/O)

# Description of the solution

## 2.1 Idea

The Idea is to use timer 1 of the PIC microcontroller. With the appropriate set up of the timer, the timer raises an interrupt on overflow. As this is 16-Bit-Timer the interrupt occurs on 65536. The timer is initialized to 65536-1000. One thousand timer counts correspond to one millisecond. So, an interrupt raised by timer one will occur every millisecond. So, the handling of the clock and the output to the ports is done by the interrupt service routine. The main program only does initialization and enters and endless loop afterwards.

## 2.2 Initialization

We separate the initialization in several parts.

For the USART data direction is set to output for Port A, B, C. Data Output LAT is initialized with 0. Serial Transmission is enabled (TXEN, SPEN) on the pins. We set Pin RD6 and RD7 as input.

Graphical user interface, text, application, email

Description automatically generated

## 2.3 Interrupt Setting

INTCON switches on interrupt handling and PIE1Bits.TMR1E enables the overflow interrupt for timer 1. T1CONbits.TMR1ON switches Timer 1 on. TMR1IF = 0 resets the interrupt flag and the statement below initialized the timer to the start value. After that it enters an endless loop.

Graphical user interface, text, application, email

Description automatically generated

## 2.4 Timer Overflow Interrupt

In the interrupt service routine, we first check if timer 1 raised the interrupt after that the interrupt flag is set to zero and after that the timer itself in reinitialized. In the following line the number of milliseconds is raised by one and it is checked if it an overflow to seconds, minutes or hours has occurred. Variables are set and reset accordingly. After that console output is generated and the output is set on the respective ports.

Text

Description automatically generated

# Ein Bild, das Text enthält. Automatisch generierte BeschreibungResults

As a result, the console regularly outputs the binary-time

Ein Bild, das Tisch enthält.

Automatisch generierte BeschreibungIO View of the used ports – Port A (Seconds) – Port B (Minutes) – Port C (Hours)

# Conclusions

To conclude, this project was very useful for all team members since everyone learned valuable skills by applying the learned knowledge from the lectures. Despite that, beginning with the project was not as simple as expected in the beginning. We had many problems and beginning with the project was a tough step for the whole group. Therefore, we seeked help from Group 7, having the same project, who helped getting the first few steps to begin with this project, such as ideas and tips, how to continue with the rest of the project. The result of the project can be seen as a success since all set requirements have been fulfilled and everyone learned valuable knowledge throughout the execution of the project.

For the future, the project could very well be expanded to add buttons with which the timer could be reset or others for example to add times to each value to match the local time, similar to a “normal” clock, where you can set the time manually (ex. 21:03:41).

# Sources

* Lessons
* StackOverflow
  + <https://stackoverflow.com/questions/14564813/how-to-convert-an-integer-to-a-character-array-using-c>
* GitHub
  + <https://github.com/AntVil/binary-clock>
* Datasheet:
  + <https://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf>
* Other Sources:
  + <https://whatis.techtarget.com/definition/USART-Universal-Synchronous-Asynchronous-Receiver-Transmitter>
  + <https://bindertronics.com/tutorials/pic-programming/buttons-as-inputs/>
  + <https://exploreembedded.com/wiki/Serial_Communication_with_PIC16F877A>